Laboratorium 7: „Tablice wielowymiarowe”

mgr inż. Leszek Ciopiński
dr inż. Arkadiusz Chrobot
dr inż. Grzegorz Łukawski

3 grudnia 2015

1. Wprowadzenie

W instrukcji 4. zaprezentowane zostały tablice jednowymiarowe. Kontynuując ten temat, niniejsza instrukcja zawiera informacje o sposobie tworzenia i korzystania z tablic wielowymiarowych w języku C.

2. Tablice

W tym rozdziale zostaną opisane zasady posługiwania się tablicami wielowymiarowymi w języku C.

2.1. Tworzenie tablic wielowymiarowych

Tablicę w języku C deklarujemy podając najpierw typ jej elementów, potem jej nazwę, a na końcu, w nawiasach kwadratowych umieszczamy liczbę jej elementów. W przypadku tablic wielowymiarowych, par nawiasów kwadratowych mamy tyle, ile wymiarów tablicy chcemy utworzyć. Deklaracja tablicy kończy się znakiem średnika. Można również tworzyć zainicjowane tablice wielowymiarowe. Przykłady deklaracji tablic podano w listingu 1. Tablice mogą być tworzone zarówno jako zmienne lokalne, jak i globalne.

```c
#define N 10    // Definicja stałej o nazwie N i wartości 10.

char tab[3][3]; // Tablica dwuwymiarowa o dziewięciu elementach (3x3) typu char.

double ulamki[N][N]; // Tablica dwuwymiarowa N×N elementowa liczb rzeczywistych.

long szescian[5][5][5];

int macierz[][3] = {{1,2,3},{4,5,6},{7,8,9}};

  // Zainicjowana tablica dwuwymiarowa (3x3). W deklaracji możemy opuścić tylko
  // jeden wymiar, pozostałe należy podać. Dodatkowe pary nawiasów klamrowych,
  // wewnątrz pierwszej pary, nie są konieczne.
```

Listing 1: Deklaracje tablic

2.2. Dostęp do elementów tablicy wielowymiarowej

W języku C, podczas odwoływania się do konkretnego pola tablicy wielowymiarowej, konieczne jest podanie w każdej parze nawiasów kwadratowych odpowiedniego indeksu. Kolejność indeksów *nie* jest przemienna. Przykłady odwołania się do tablicy wielowymiarowej podczas odczytu i zapisu zaprezentowano na listingu 2 zawierającym kilka przykładów.

```c
#include <stdio.h>

#define WIERSZ 10
#define KOLUMNA 5

int main(void)
{
    int i, j;
    char macierz[WIERSZ][KOLUMNA];

    //wpisanie danych do macierzy
    for(i = 0; i<WIERSZ; i++)
        for(j=0; j<KOLUMNA; j++)
            macierz[i][j]=i*10+j;
    // odczyt danych z macierzy
    printf("\n[\n");
    for(i = 0; i<WIERSZ; i++){
        for(j=0; j<KOLUMNA; j++)
            printf("%d\t",macierz[i][j]);
        printf("\n");
        }
    printf("]\n");


    return 0;
}
```

Listing 2: Sposoby dostępu do elementów tablicy wielowymiarowej

W wyniku działania programu w konsoli uzyskujemy napis:

```
[
0  1  2  3  4  5  6  7  8  9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
]
```

3. Funkcje, a tablice wielowymiarowe

Relacje pomiędzy funkcjami, a tablicami wielowymiarowymi zaprezentowane będą w oparciu o listing 3.

```c
#include <stdio.h>
#define WIERSZ 10
#define KOLUMNA 5
typedef char mac[WIERSZ][KOLUMNA];

void uzupelnij(char macierz[WIERSZ][KOLUMNA])
{
    int i,j;
    for(i = 0; i<WIERSZ; i++)
        for(j=0; j<KOLUMNA; j++)
            macierz[i][j]=i*10+j;
}

void uzupelnij2(char (*macierz)[KOLUMNA])
{
    int i,j;
    for(i = 0; i<WIERSZ; i++)
        for(j=0; j<KOLUMNA; j++)
            macierz[i][j]=i*10+j+50;
}

void wyswietl(char macierz[][KOLUMNA])
{
    int i,j;
    printf("\n[\n");
    for(i = 0; i<WIERSZ; i++){
        for(j=0; j<KOLUMNA; j++)
            printf("%d\t",macierz[i][j]);
        printf("\n");
        }
    printf("]\n");
}

int wieksza_suma(mac m1, mac m2)
{
    char i, j;
    int suma1=0, suma2=0;
     for(i = 0; i<WIERSZ; i++)
        for(j=0; j<KOLUMNA; j++){
            suma1+=m1[i][j];
            suma2+=m2[i][j];
        }
    if(suma1>suma2) return -1; //return kończy funkcje
    if(suma1<suma2) return  1; //return kończy funkcje
    return 0;
}

int main(void)
{
   char m[WIERSZ][KOLUMNA];
   mac m2;

   uzupelnij(m);
   uzupelnij2(m2);

   switch (wieksza_suma(m, m2)){
        case -1: wyswietl(m); break;
        case  1: wyswietl(m2); break;
        default: printf("Sumy elementów obu macierzy są równe");
   }

   return 0;
}
```

3.1. Parametry

Tablice przekazywane są wyłącznie przez wskaźnik. Oznacza to, że zmiany wykonane na tablicy wielowymiarowej wewnątrz funkcji zawsze pozostają widoczne po jej zakończeniu. Parametr tablicowy tworzony jest tak jak deklaracja tablicy wielowymiarowej, ale jedną parę nawiasów kwadratowych wolno zostawić nam pustą (wiesz 22). Podanie wszystkich rozmiarów tablicy nie wpływa na działanie funkcji, ale może zwiększyć czytelność jej kodu (wiersz 6). Możliwe jest też zadeklarowanie typu tablicowego (wiersze 4 i 34). Jeszcze inny przykład deklaracji parametru jako tablicy dwuwymiarowej z wykorzystaniem wskaźnika przedstawiono w wierszu 14. Umieszczenie słowa kluczowego `const` nie czyni przekazywanej tablicy odporną na modyfikacje.

3.2. Zwracanie wartości

Funkcja nie może zwracać wartości będącej tablicą. W celu porównania sumy wszystkich pól dwóch macierzy zdefiniowano w programie *kod błędu*. Zadaniem takiego kodu jest zwrócenie informacji o wyniku wykonanego działania, zarówno o niepowodzeniu, jak i powodzeniu. W przedstawionym przypadku, funkcja informuje, w której macierzy suma wszystkich elementów jest większa (wiersze 43–45). Taki kod wymaga jeszcze odpowiedniej interpretacji (wiersze 56–60). Jako wynik, wyświetlone zostaną wartości od 50 do 99.

4. Zadania

Uwaga! Wszystkie programy muszą być napisane z podziałem na funkcje z parametrami.

1. Napisz program, który wykona mnożenie macierzy o wymiarach 4×3 przez skalar[1]. Macierz należy wypełnić liczbami typu `double` losowanymi z przedziału $(-11, 11)$ i wypisać przed i po pomnożeniu. Skalar powinien być wprowadzony przez użytkownika.

2. Napisz program, który znajdzie największą i najmniejszą wartość w macierzy liczby typu `int` o wymiarach 3×4. Macierz należy wypełnić liczbami całkowitymi losowanymi z zakresu od -5 do 5 i wypisać jej zawartość na ekranie.

3. Napisz program, który posortuje macierz o wymiarach 4×4, wypełnioną liczbami typu `int` losowanymi z zakresu od -10 do 10. Zawartość macierzy należy wypisać na ekran po jej wypełnieniu.

4. Napisz program, który wypełni macierz o wymiarach 4×4 liczbami całkowitymi losowanymi z zakresu od -20 do 20, wypisze jej zawartość na ekran, a następnie wypisze sumy elementów należących do poszczególnych wierszy i kolumn tej macierzy.

5. Napisz program, który wypełni macierz o wymiarach 5×5 liczbami naturalnymi losowanymi z zakresu od 6 do 25, następnie wypisze jej zawartość na ekran, a potem policzy różnicę sum wartości elementów leżących na przekątnej i przeciwprzekątnej tej macierzy.

6. Napisz program, który wypełni macierz o wymiarach 5×5 liczbami naturalnymi losowanymi z zakresu od 0 do 15, wypisze zawartość tej macierzy na ekran, a następnie odwróci kolejność liczb leżących na przekątnej i ponownie wypisze zawartość tej macierzy na ekran.

7. Powtórz poprzednie zadanie dla liczb leżących na przeciwprzekątnej macierzy.

8. Napisz program, który pozwoli użytkownikowi zapisać w tablicy łańcuchów znaków 10 wyrazów, wypisze na ekran zwartość tej tablicy, a następnie znajdzie wyraz najdłuższy i wypisze go na ekran.

[1] Skalar, to inaczej liczba. Należy przez nią pomnożyć wartość każdego elementu macierzy.